Sprite Generation Using Sprite Fusion

Y. CHEN, A.A. DESHPANDE, AND R.S. AYGÜN* Computer Science Department, University of Alabama in Huntsville

There has been related research for sprite or mosaic generation for over 15 years. In this paper, we try to understand the methodologies for sprite generation and identify what has not actually been covered for sprite generation. We firstly identify issues and focus on the domain of videos for sprite generation. We introduce a novel sprite fusion method that blends two sprites. Sprite fusion method produces good results for tracking videos and does not require object segmentation. We present sample results of our experiments.

Categories and Subject Descriptors: I 2.10 **[Artificial Intelligence**] – Vision and Scene Understanding – *Video Analysis.*

General Terms: video processing

Additional Key Words and Phrases: Sprite generation, video processing, video standards

ACM Reference Format:

CHEN, Y., DESHPANDE, A.A., AND AYGÜN, R.S. 2011. Sprite Generation Using Sprite fusion. *ACM Trans. Multimedia Computing, Communications, and Applications*, X, X, Article X (X 2011), 20 pages.

DOI=X

1. INTRODUCTION

Sprite (or mosaic) generation plays an important role in object-based coding, video compression, security, object recognition, and behavior analysis. The sprite generation has gained significant importance as the introduction of MPEG-4 Visual Standard [MPEG4-2; Sikora 1997], since it provides efficient video compression and interactivity with objects. The latest standard for video compression is H.264 / Advanced Video Coding (AVC) MPEG-4 Part 10 [H264; Ostermann 2004] while upcoming standard, H.265 [H.265; 2009] is being developed. It has been shown experimentally that H.264/MPEG4-10 provides better compression (around 50% bit savings) than MPEG4-2 [Ostermann 2004]. This may lead to the misconception that H.264 will replace MPEG4-2 at all levels. In fact, the comparison tests usually evaluate H.264 against MPEG-4 Simple Profile. In [To 2005], it has been shown that when sprites are encoded using JPEG-2000 Region-of-Interest [Taubman 2002] and motion-compensated predicted image differences are coded with H.264, the compression and quality of sprite coding is better than the compression and quality of H.264. Sprite coding with H.264 has also been studied in recent research [Kunter 2008]. The sprite coding only exists in the Main Visual profile of MPEG-4. Although there are (commercial) MPEG-4 encoders available, to the best of our knowledge none of them implements the Main Visual Profile.

1.1 Issues

There are crucial issues related to sprite generation: accuracy, performance, robustness, and domain. *Accuracy.* Accuracy (or quality) of sprite is measured in two phases in general: subjective and objective. In subjective evaluation, an expert decides whether the sprite is a correct sprite for the video or not. In objective evaluation, an objective measure such as Peak Signal-to-Noise Ratio (PSNR) is computed for frames that are regenerated from the sprite against the original frames. In [Chen 2010], we used synthetic

* This material is based upon work supported by the National Science Foundation under Grant No. 0812307.

Authors' addresses: Y. Chen, Computer Science Department, University of Alabama in Huntsville, Huntsville, AL, USA E-mail: yichen@cs.uah.edu ; A. Deshpande, Computer Science Department, University of Alabama in Huntsville, Huntsville, AL, USA E-mail: adeshpan@cs.uah.edu ; R.S. Aygün, Computer Science Department, University of Alabama in Huntsville, Huntsville, AL, USA E-mail: raygun@cs.uah.edu

DOI http://dx.doi.org/10.1145/2168996.2169002

videos to check the accuracy of sprite generation. *Performance.* There is inherent complexity of sprite generation since it includes three major steps: global motion estimation, warping, and blending. The sprite

generation process has limitations due to: a) motion models, b) scene depth, and c) moving objects. The algorithms to overcome these limitations should not introduce significant burden on the system since sprite generation could be part of an encoder. *Domain*. The features that affect the domain are the presence of global motion, magnitude of global motion, presence of moving objects, and mobility of these objects. The domain is related to the identification of videos suitable for generating sprites. *Robustness.* If a sprite can be generated for a domain of videos, a robust sprite generation algorithm should be able to generate sprites for all videos in the domain not just a set of specific videos.

The most general way to detect the camera motion is to detect the global motion between consecutive frames. If an appropriate motion model is used, the scene depth can also be described properly. However, there is no clear and robust way to ignore moving objects during sprite generation. Under (background) sprite generation context, moving objects are considered as foreground and the rest are considered as the background. The most general approach is that the long-time aggregation of aligned pixel values from frames by eliminating pixels that yield error over a threshold will remove the moving objects from the sprite. However, these long-term aggregation methods cannot eliminate the moving objects thoroughly, and thus, produce inaccurate sprites. The statistical approaches like Kalman filter are able to reduce the effect of moving objects. The object occlusion problem worsens the object segmentation problem. The sprite coding cannot be done in real time because of object segmentation [To 2005]. The limitations can be summarized as: no object removal; conditional object removal; necessity of segmentation masks; necessity for (rough) foreground segmentation; necessity of all frames; temporal integration, and visually improper sprite generation.

1.2 Related Work

There are various definitions for sprites, background, and mosaic. The (background) sprite (or mosaic) can be defined as the stable background where the moving objects are replaced with the original background pixels. The mosaic generation process requires the integration of background pixels that are spanned over frames. The sprite is not only generated for the background but it can be generated for any object.

Sprite generation research has been studied under panoramic image, salient stills [Teodosio 1993], mosaic, background extraction, sprite coding, and image alignment-stitching [Szeliski 2006]. Sprite generation has 3 steps: global motion estimation (GME) [Dufaux 2000; Smolic 1999; Smolic 2000; Lu 2001; Lee 1997], warping, and blending.

Sprite (mosaic) generation methods mostly differ in the ways motion estimation and features used to create the mosaic. In [Irani 1998], different representations (or types) of mosaics like static, dynamic and synopsis mosaic have been investigated. Depending on camera motion, different types of mosaic are generated: planar mosaic [Irani 1998; Szeliski 1997; Smolic 1999; Smolic 2000; Zoghlami 1997; Salembier 1998], the cylindrical mosaic [Zoghlami 1997], and the spherical mosaic is generated when camera is both panning and tilting [Coorg 2000]. Alternative methods such as pipe projection [Peleg 2000] and salient stills [Teodosio 2005] are proposed to deal with forward motion and zoom, respectively. Multiple static cameras [Geys 2006], multi-sprites [Farin 2006], and high resolution mosaic [Smolic 2000] are various techniques for mosaic or sprite generation. Work on estimation of motion parameters and generation of sprites has been presented in [Smolic 1999]. The blurring that may be caused by inaccurate segmentation masks is dealt with in [Lu 2003]. In our prior work [Aygun 2002], we developed a high-resolution mosaic (sprite) to reduce blurring in the sprite. We introduced a histemporal filter for blending. In [Lai 2009], spatial correlation is used to further increase the accuracy since most probable values for a region may not actually be spatially correlated. There is also some research that use additional resources such as availability of multiple views or multiple shots etc. Some applications target coding and they just try to reduce the number of bits for compression.

Most of the test videos (except standard test sequences) are not available to the community. Table I provides a summary of sequences that are used in the literature. The limitations can be summarized as: a) no object removal; b) conditional object removal; c) necessity of segmentation masks; d) necessity for (rough) foreground segmentation; e) necessity of all frames; f) temporal integration; and g) visually improper sprite generation. Some applications do not deal to remove moving objects (or they target object-free videos). Sometimes applications require that last or first frame should not have objects for total object

Table 1 Sequences used in the literature

References	Sample Video sequences used	Limitations
[Lu 2003]	Stefan, Coastguard	Applies rough foreground segmentation; partial coastguard sprite
[Cheung 2008]	Stefan	Relies on availability of segmentation masks
[Shen 2004]	Coastguard	Rough foreground segmentation; partial coastguard sprite
[Lu 2001]	Stefan, Coastguard, Foreman	Rough foreground segmentation; no coastguard sprite
[Aygun 2002]	Coastguard, Foreman	No object removal
[Cherng 2007]	surveillance video like hall qcif	Temporal integration; no coastguard sprite
[Ye 2005]	Stefan, Mobile Calendar	Requires availability of segmentation masks
[Grammalidis 1999]	Claude	Multiview is required
[Cheung 2007]	Stefan, Coastguard, Foreman, ZoomIn	Relies on availability of segmentation masks
[Krutz 2008]	Stefan, Race-1	Temporal integration; no coastguard sprite
[Asif 2008]	Lab Sequence, Outdoor Sequence	Applies rough foreground segmentation
[Ye 2008]	Stefan	Computes overlaps with every frame in a cluster
[Aygun 2004]	Lecture Video	No object removal
[Kunter 2008]	Stefan, Mountain Sequence	Correct sprite is not an issue
[Krutz 2006]	Stefan, Mobile Calendar, Horse Sequence	Temporal integration; requires availability of all frames for determining middle frame ; no coastguard sprite
[Zhu 1999]	Library, Main building sequence	Objects removed only if last frame does not have objects
[Teodosio 1993]	video zoom featuring the cellist Yoyo Ma. Also referred as Ma sequence	Temporal integration; no coastguard sprite
[Dasu 2004]	Stefan	Temporal integration; no coastguard sprite
[Parikh 2007]	Aerial Video over a Desert	No object removal
[Steedly 2005]	AC1, AC5, AC8, AGP2, GP1, GP4, GP5, WF, LK	No object removal
[Marzotto 2004]	S. Zeno, Map Of Europe	No object removal
[Bevilacqua 2005]	S1, S2 (indoor panning), S3 (Outdoor strong and with changes in depth)	Applies clustering to determine foreground & background pixels; likely to fail for tracking videos
[Hsu 2004]	Coastguard, Golf, Festival, Basketball, Football-1, Football-2	Requires object segmentation; computation intensive
[Dufaux 2000]	Stefan, Coastguard, Foreman, Table tennis, MIT, Coast_shore	No object removal
[Smolic 1999]	Stefan, Foreman, Mobile Calendar, Bus, Horse	Removes objects only if first frame does not have objects
[Smolic 2000]	Stefan	Removes objects only if first frame does not have objects
[Irani 1998]	Baseball, Airport, Flying plane, Parachuters	Temporal integration; no coastguard sprite
[Teodosio 2005]	Ma sequence, Fastball, Basketball, Football, Street surveil., Gymnastics	Temporal integration; no coastguard sprite
[Farin 2006]	Stefan, table tennis, Rail	Requires availability of all frames; not real-time
[Chen 2006]	Skater	Requires multiple similar shots; low-quality sprite
[Salembier, 1998]	Coastguard	Improper object elimination
[Lai 2009]	Tennis	Spatio-temporal integration; no coastguard sprite

removal. Some applications assume that segmentation masks are already available. Rough foreground segmentation relies on local object motion different from global motion. This might fail as objects start to act static. Almost-reliable foreground segmentation is also computation intensive. Some applications assume

the availability of all frames at once. This limits the real-time use of the method. Temporal integration methods basically assume that frequency of background pixel is dominant over the frequency of a foreground pixel at a specific location. They fail for videos such as 'coastguard' video. All temporal integration methods either avoid using coastguard video or just show partial coast-shore of the video. They cannot cope with the pattern of the boat in the video. In some research, we have observed that sprite is basically (visually) not correct and full of artifacts. Sample problems are significant blurring, shadows or ghosts of objects, missing parts on sprite, etc.

1.3 Our Approach

In this paper, we firstly classify the videos into six classes based on moving object presence, object displacement, camera motion, and object location in a frame. We show that in some videos (tracking) even though they may have moving objects or object occlusion, it is possible to generate sprite without any object segmentation. If moving objects follow a pattern, there is no need to segment objects. We describe our pattern informally as: "If moving objects do not appear at the border of frames, the sprite can be generated without using object masks or detection of object pixels". We call such videos as tracking videos since the camera is tracking objects.

We introduce the sprite fusion that is able to generate the sprite by ignoring the moving objects without any segmentation and detection of object pixels. Our sprite fusion method employs merging on two types of sprites: *assertive* and *conservative* sprite. In assertive sprite generation, a pixel from a new image is integrated into the sprite whether a previous pixel was integrated onto that location or not whereas in conservative sprite generation a pixel is only integrated if no pixel was integrated to that location before. Our fusion is different from fusion techniques for semantic multimedia information retrieval [Atrey 2010]. These techniques fuse unimodal features obtained from each modality. These are usually classified as early or late fusion depending on when learning is applied [Snoek 2009]. We fuse assertive sprite onto conservative one or vice versa. In each sprite there is a region that contains the object. However, the other sprite does not have the object at that location. If pixels of a region containing object are replaced with the corresponding pixels in the other sprite, the objects can be removed. Conservative sprite has the object in the region of first frame whereas assertive sprite has the object in the region of the last frame.

For each video, two sprites are generated and then integrated to yield the final sprite of the video. In our method, as long as objects do not appear at the borders of a frame and the scene in the last frame is different from the scene in the first frame, it is possible to generate sprite without any object segmentation. We experimented our methods on standard test sequences like *coastguard* and *stefan*. To the best of our knowledge, sprite fusion is the first method that can ignore moving objects without object segmentation or detection of object pixels during sprite generation. We have developed an open database of videos that are accessible through http://sprite.cs.uah.edu/. Interested users may access the database and perform their experiments. In addition, we also present results on a set of challenging videos for sprite generation. Our goal in this paper is not to increase the accuracy of sprite generation. If accuracy can be computed, it means there are already some algorithms that can generate the sprite at an acceptable level. Our goal is to generate sprites for videos which are difficult to handle. Since we target videos that sprites were not generated in the past, our research is not incremental but a transformative one. We explain the challenges and future directions for sprite generation.

Our contributions can be summarized as follows:

- Increasing domain of videos where sprite can be generated,
- Sprite generation *without object segmentation* for partially-tracking and tracking videos, and
- Introducing the novel sprite fusion technique for partially-tracking and tracking videos by fusing assertive and conservative sprites.

This paper is organized as follows. The following section explains the classification of videos for sprite generation. Our sprite generation methods are described in Section 3. Our experimental results are provided in Section 4. The last section concludes our paper.

2. CLASSIFICATION OF VIDEOS BASED ON MOTION INFORMATION

To determine which domain a video belongs, the videos need to be classified. Firstly, we provide features for video classification and then explain how videos are classified.

2.1 Features for Video Classification

It is necessary to classify videos based on camera motion, presence of objects, object displacement, and object appearance at the borders of a frame. Sprite coding can benefit from the sprite if there is a significant camera motion. The objects in the foreground should be eliminated. To generate the background behind an object, the object should displace its location. Objects at the borders of a frame may aggravate object elimination.

Our feature set for classifying videos for sprite generation is composed of 5 features: *{G, C_g, N, P, M}* [Deshpande 2009]. These correspond to global motion *(G)*, cumulative global motion *(C_g)*, the number of objects in motion *(N)*, presence of objects at the borders of a video frame *(P)* and the number of macroblocks in motion (*M*). We briefly explain these features.

2.1.1 *Global Motion and Cumulative Global Motion. Global motion parameters (G).* The global motion estimation is the first step of sprite generation. The perspective motion has 8 parameters and the new coordinates of a pixel at (x, y) is computed as:

$$x' = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1} \qquad y' = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} \tag{1}$$

where (*x', y'*) is the position of the same pixel in the reference frame (i.e., next frame).

Cumulative maximum global motion (C_g). Consider Fig. 1 for cumulative global motion. The pixels that are marked correspond to the same locations with respect to the size of a frame. The distance between them on the sprite indicates the cumulative global motion between frames 0 and 299 with respect to the selected pixel. Note that generating sprite is not needed for this feature. The sprite is used merely to explain the concept.

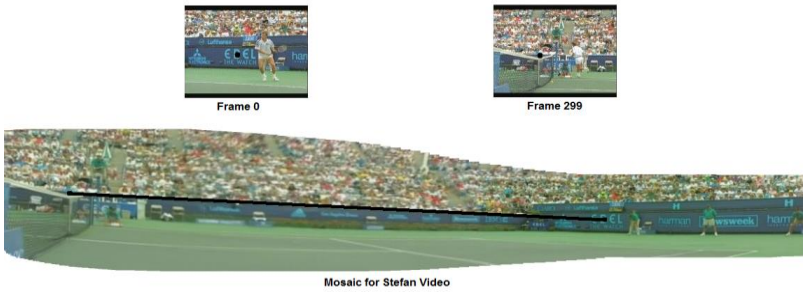


Fig. 1. Cumulative Global Motion for a selected point

2.1.2 *Objects.* In order to extract the features like *the number of objects in motion, the number of macroblocks in motion* and *presence of object at the border of the image,* we compute the local motion present in the video. The local motion is subtracted from the global motion. We estimate the following features based on the local motion.

Number of objects in motion (N). We eliminate the macroblocks that do not have any neighboring macroblock having the motion. The number of the objects is estimated by region growing algorithm from macroblocks having motion. Note that we do not require an accurate estimation of the number of objects.

Presence of objects at the borders of a frame (P). We also determine whether an object appears at the borders of a frame or not. This helps us identify the type of video. Especially, in tracking videos, the objects are maintained close to the center of a frame.

Number of macroblocks in motion (M). This feature counts the number of macroblocks that have motion. The number of objects might be misleading if there are multiple moving objects that have a neighboring macroblock with another object. All these features are identified for each frame in the video.

2.2 Video Classes

In this section, we first describe the general classes and then focus one of the classes.

2.2.1 *General Video Classes.* We have determined six classes based on these four properties. These classes can be briefly described as follows:

- *Static Video (StV).* Any frame of a video is equivalent to an image (and also sprite) captured by a static camera from a scene without objects.
- *Scenery Video (ScV).* The scenery class does not have moving objects in the videos. Based on the structure of the scene, the sprite can be generated based on selected motion model that defines the scene. For example frames 230 through 300 of 'foreman' sequence belongs to scenery class.
- *Commercial-like Video (CommV).* This class of videos usually does not have any significant camera motion. Even in the presence of global motion, it may be hard to detect any pattern in the global motion. There may also appear some objects.
- *Earthquake-like Video (EV).* This class of videos does not have any significant camera motion. The camera might be shaken due to natural events or by the cameraperson. It may be possible to detect a pattern in the motion of the camera. There might be some moving objects in the scene.
- *News, Educational and Surveillance like Video (NES).* The videos in NES class have objects that do not change their locations. Therefore, the background behind the objects is not visible, and it may not be possible to generate the sprite. News (or anchor) videos are in this class. MPEG test sequence 'akiyo' belongs to this class. If a video has too many moving objects, even though objects in the video displace their locations, the background may not be visible because of occlusion. Videos captured from a camera that monitors a very crowded street fits to this category.
- *Complex Video (CompV).* In complex videos, the camera might be tracking multiple objects or there might be more than one object but the camera is tracking one of them. Some sports applications like soccer fall under this category. MPEG test sequence *coastguard* belongs to this class. The tracking subclass emphasizes that there should not be any moving object at the borders of a frame. In other words, there is an object being tracked. Some sports videos like tennis fall under this category. The MPEG test sequence *stefan* belongs to this class.

The most representative video type is used for naming the classes. For example, *earthquake-like* class does not have only earthquake videos, but it is named so because the earthquake video best represents this class. We use a decision-based classifier using these features to classify videos. In addition, we also tried to determine the number of frames that can help to determine the type of a video. Our accuracy for video classification is 82% [Deshpande 2009]. Further information about the dataset and classification can be found in [Deshpande 2009].

2.2.2 *Tracking and Partially-Tracking Videos.* The complex videos can be further classified as *tracking* and *crowded* classes. In *crowded* class, the camera may be tracking a set of objects but there may be also many other moving objects that may enter the field of view. It may be impossible to generate the background sprite for the crowded class. Therefore, we focus on tracking videos. Before giving a formal definition on tracking videos, we would like to define *aligned frame difference* and *intersection*. Assume that there are n frames in the sequence where f_1 and f_n are the first and last frames in the sequence.

Aligned frame intersection. The aligned frame intersection of frames f_i and f_j is represented as $f_i^j \cap f_j$ and corresponds to the overlapping area after frames are aligned on top each other where f_i^j represents the alignment of f_i over f_j without integration.

Aligned frame difference. The aligned frame difference of frames f_i and f_j is represented as $(f_i^j - f_j)$ and corresponds to difference of frame f_i from f_j after frames are aligned where f_i^j represents the alignment of f_i over frame f_j without any integration. For a pixel p, $p \in (f_i^j - f_j)$ if and only if the following case occurs (Fig. 2):

$$(p \in f_i^j) \wedge (p \notin f_j)$$

Tracking Videos. Let $O = \{o_1, o_2, ..., o_m\}$ represent the set of moving objects in the video where m represents the number of moving objects. The cardinality of O is represented with |O|. We do not detect or extract moving objects, but we use it to explain the concept. In tracking videos, the camera tries to locate the target object in the middle of a frame. The goal is to keep the object away from the borders of frames. Theoretically, a video is considered to be a tracking video if

$$not\ (\exists i \exists k\,(o_k \in O \wedge o_k \in (f_i^{i-1} \cap f_{i-1}))) \text{ where } \quad 2 \le i \le n\ .$$

In other words, objects should not appear in the new areas.

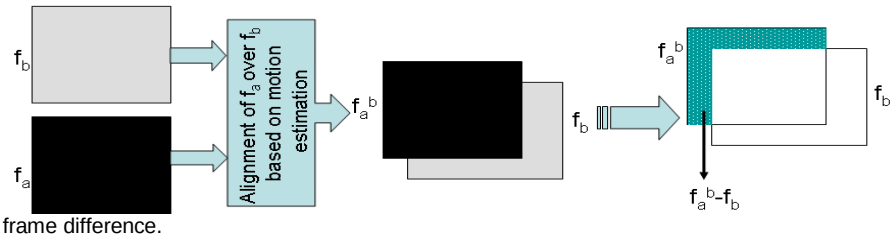


Fig. 2. The aligned frame difference.

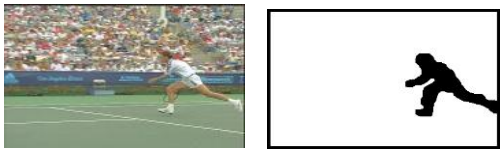

(a) (b)

Fig. 3. Example of an object at borders of a frame a) frame 255 of stefan, b) segmentation mask for frame 255.

Partially-Tracking Videos. Based on the definition of tracking videos, any video that has objects in the frame differences is not considered as a tracking video (Fig. 3). The appearance of objects at the borders is not a problem as long as the border having moving object is covered in a previous frame. Also note that it is permissible for an object to appear as long as it is not in the aligned difference. This type of videos is classified as partially-tracking videos. In this type of videos, the cameraperson cannot maintain the target object completely in some of the frames of the video. The partially-tracking videos can be expressed as follows:

$$not\ (\exists i \exists k\,(o_k \in O \wedge o_k \in f_i^{\,i-1} \wedge (f_i^{\,i-1} - \bigcup_{m=1}^{i-2} f_{m+1}^{\,m}) \neq \phi)) \text{ where } \quad 2 \le i \le n\ .$$

Figure 3 (a) and (b) shows the frame 255 and the corresponding object mask of *stefan* sequence where the player appears at the border. Stefan test sequence is actually partially-tracking video.

2.2.4 *Relevance for Classes for Sprite Generation.* We analyze videos in two parts: eligibility for sprite generation and contribution to compression through sprite coding (Table II). For sprite generation, we inspect whether all components of the background are visible some time throughout the video. We look into two conditions to analyze the usefulness of sprite generation for compression: presence of significant camera motion and motion of objects such that background behind the objects becomes visible.

TABLE II. Classes and Suitability for Sprite Generation & Coding

Class Name	Sprite Generation	Sprite Coding
Static video	Yes	Not advantageous
NES-like video	Mostly No	Not advantageous
Earthquake-like	Yes	Not advantageous
Commercial-like	Mostly No	Not advantageous
Scenery videos	Yes	Advantageous
Complex videos	Yes for tracking videos	Advantageous

3. SPRITE GENERATION

The sprite generation has three steps: global motion estimation (GME), warping, and blending. Since GME is the most costly step [MPEG4 Software; Nagaraj 2001], GME is performed once per frame. It is likely that GME between frames may yield error. The accumulation of errors due to frame-to-frame GME may yield poor sprites [Smolic 1999]. We extract the previous frame from the sprite and apply GME on the extracted frame and current frame as in [Smolic 1999]. This avoids accumulation of errors.

Let S be the sprite where the frames are warped and blended into it. All pixel values of S are assigned to 0 initially. Let *S(i, j)* denote the pixel value at location *(i, j)* of the sprite. Let *f(i', j')* be the corresponding pixel on frame *f*. However, because of real numbers produced in GME, *i'* and *j'* are unlikely to be integer values. To estimate *f(i', j')*, we use bilinear interpolation as follows:

$f(i', j') = (1 - b) * (1 - c) * f(p, q) + b * (1 - c) * f(p, q + 1)$
$+b * c * f(p + 1, q + 1) + (1 - b) * c * f(p + 1, q);$

where $p = \lfloor i' \rfloor, q = \lfloor j' \rfloor$, $b = i' - p$, and $c = j' - q$.

We propose two new types of sprite termed as *conservative* and *assertive* sprites.

3.1 Conservative and Assertive Sprite Generation

Conservative Sprite. In conservative sprite generation, a pixel from the new (or consecutive) image is integrated into the sprite if no pixel was integrated onto that location before. The temporal integration for conservative sprite is as follows:

$$S(i', j') = f_k(i', j') \text{ if } S(i, j) = 0.$$

Figure 4 shows the conservative sprite generated from frames 226 to 245 of *stefan* video.

Fig. 4. Conservative sprite generated from frames 226 to 245 of 'stefan'.

Fig. 5. Assertive sprite generated from frames 226 to 245 of 'stefan'.

Assertive Sprite. In assertive sprite generation, a pixel from the new (consecutive) image is integrated into the sprite whether a previous pixel was integrated onto that location or not. In dynamic sprite generation [Irani 1998], the sprite is generated with traditional temporal integration methods and then the last frame is integrated into the sprite. In terms of temporal integration, the assertive sprite generation is different from the dynamic sprite. The temporal integration for assertive sprite is: $S(i, j) = f_k(i', j')$

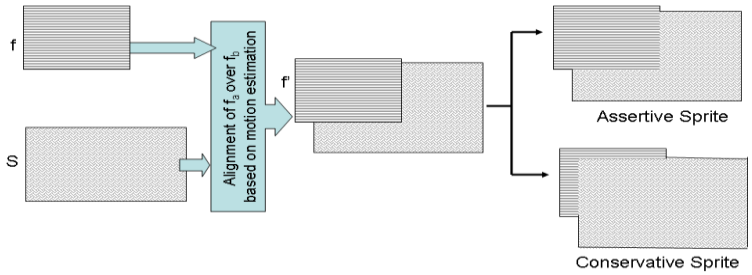


Fig. 6. Assertive and conservative sprite generation.

Figure 5 shows the assertive sprite generated from frames 226 to 245 of 'stefan' video. The differences between conservative and assertive sprite generation is depicted in Figure 6.

3.2 Sprite Fusion

We propose the sprite fusion technique by merging two types of sprites: assertive and conservative. In our system, we keep the motion parameters based on the first frame in the sequence. During sprite generation process, we need to know where the first frame is located. As long as the camera is panning right and/or tilting down, the location of the initial frame on the sprite does not change. The location on the sprite only changes if the camera is panning left or the camera is tilting up. To integrate the new areas, the sprite needs to be shifted right or down. In that case, the location of original frame is relocated on the sprite image.

3.2.1 *Fusing assertive sprite onto Conservative Sprite.* Initially, the origin of the initial frame is located at *(width/2, height/2)* of the sprite where width and height are the width and height of the frames, respectively. If a shift is required after motion estimation, the origin of the initial frame is moved to $(width/2 + shift_x, height/2 + shift_y)$ where $shift_x$ and $shift_y$ represent the shift in the horizontal and vertical directions, respectively. Let $shift_x[i]$ and $shift_y[i]$ be the shifts in horizontal and vertical direction after motion estimation between frames f_i and f_{i-1}, respectively. At the end of the sprite generation,

$$shift_x = \sum_{2}^{n} shift_x[i] \quad shifty = \sum_{2}^{n} shift_y[i]$$

Eventually, the top-left corner of the initial frame can be located as *(shift$_x$, shift$_y$)*. Notice that the conservative sprite maintains the initial frame on the sprite. The area of initial frame in conservative sprite is replaced by the corresponding area in the assertive sprite. Figure 7 displays where the initial frame 226 in conservative sprite (Figure 4) is replaced with the corresponding area from assertive sprite (Figure 5).

3.2.2 *Fusing Conservative Sprite onto Assertive Sprite.* The fusion on assertive sprite needs to be handled carefully. The assertive sprite maintains the last frame on the sprite. However, the last frame is warped onto the sprite based on the motion parameters obtained after motion estimation. Therefore, the last frame on the sprite is unlikely to have the size and borders of the original last frame.

Although the replacement of the region of the last frame looks simple, the detection of all pixels must be handled carefully. One way is actually to replace the pixels in a (affine) region with the pixels of the conservative sprite. This requires the scanning of pixels (some of them outside the region) and then determining whether a pixel resides in the region. However, this method continuously checks whether a pixel is in the frame or not.

Fig. 7. Fusion on conservative sprite for frames 226 to 245 of 'stefan'.

Fig. 8. Fusion on assertive sprite for frames 226 to 245 of 'stefan'.

The alternate method (the one we use) is to apply sprite fusion during warping the last frame onto the sprite. In this case, we generate a sprite that is equivalent to the assertive sprite. However, when warping the new frame, we rather get values from the conservative sprite and then incorporate those new values. The algorithm for sprite fusion on assertive sprite is given in Algorithm 1. Figure 8 displays where the initial frame 245 in assertive sprite (Figure 5) is replaced with the corresponding area from conservative sprite (Figure 4).

Algorithm 1. Algorithm for fusion on assertive sprite

```
// IN: Assertive sprite AS_{n-1}
// IN: Conservative sprite CS_{n-1}
// IN: frames f_n and f_{n-1} of the video
// OUT: Fused sprite FSA on assertive sprite
estimate motion parameters M_n between frames f_n and f_{n-1}
apply the necessary shift operations on AS_{n-1} and CS_{n-1}
copy AS_{n-1} to FSA
for i = 1 to height do
  for j = 1 to width do
    compute the estimated pair (i′,j′)on frames based on M_n
    AS(i, j) = f_k(i′, j′)
    if CS(i, j) = 0 then
      CS(i, j) = f_k(i′, j′)
    end if
    FSA(i, j) = CS(i, j)
  end for
end for
```

3.3 PROOF

Our claim is that (informally) if the object does not appear at the frame borders, the sprite can be generated without object segmentation. We assume that global motion is estimated correctly. When two frames are overlapped where the first frame is warped onto the second frame, there may be new areas that were not included in the sprite (Figure 9). Assume that we apply conservative sprite, and the shaded areas (region *A*) are new regions that are not covered in the previous frames (Figure 10).

General Idea. We are going to prove our claim in two steps using Figure 10: a) region *A* can be generated without objects, and b) region *B* can be generated without objects. We use conservative sprite to generate region *A* without objects. We assume that the new regions that are contributed by a frame do not have objects. Fig. 9 shows 4 (shaded) regions (that was not covered by the previous frame) where the new frame contributes to the sprite: R_1, R_2, R_3, and R_4. Possible overlappings for two rectangular frames are provided in the Appendix A.1. The sprite is expanded by using these 4 regions as long as these regions are not covered in the sprite. Since sprite expansion is based on regions that do not have moving objects, region *A* can be generated without objects.

Assume that there is frame (f_x) that corresponds to an area in region *A* but does not intersect with region *B*. Region *B* corresponds to the region of first frame in the sequence. By using the same logic but starting from frame f_x, we expand the sprite again without objects. Region *B* is in the expanded area and it does not have moving objects.

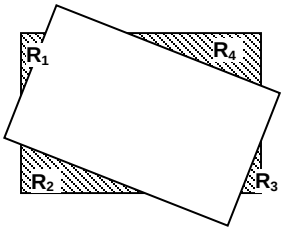


Fig. 9. New areas for sprite

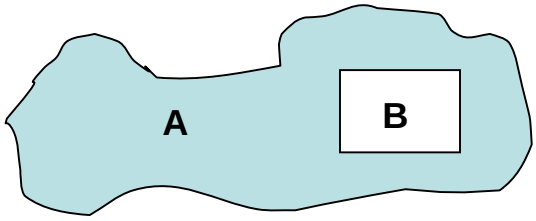


Fig. 10. Sprite is equivalent to (*A U B*)

We do not want to make two passes for sprite generation: one pass for region *A* (in forward direction) and one pass for region *B* (in reverse direction). We are going to show that conservative sprite in reverse direction is equivalent to assertive sprite in forward direction. Thus, region *A* and region *B* can be generated without objects simultaneously. These three steps will complete our proof.

Notation. We are going to use different notations for the purpose of proofs. ¤*(R)* represents a predicate that indicates the absence of moving object in region *R*. Let f_i represent the i^{th} frame in a video *(0≤i≤n)*. S_i represents the sprite after *i* frames are warped and blended. Let $\hat{f}_i$ represent the frame after warping onto sprite S_{i-1}. $\bar{f}_i$ represents the region of frame f_i on the final generated sprite S_n whereas $\tilde{f}_i$ represents the alignment of frame f_i on the previous frame f_{i-1}. The new areas that are presented in a frame with respect to a previous frame is represented by $\Delta_i = \tilde{f}_i - f_{i-1}$. Let $\delta_i = \hat{f}_i - S_{i-1}$ and represent the new regions of f_i to be blended onto S_{i-1}. Note that $\delta_i \subseteq \Delta_i$ since $S_{i-1} \supseteq \hat{f}_{i-1}$. If sprite is generated by processing frames in reverse order: S_n is the first sprite after processing the first frame f_n; $\hat{f}_i$ represents the frame after warping onto sprite S_{i+1}; and $\bar{\delta}_i = \hat{f}_i - S_{i+1}$.

Constraint 1. $\exists i(\bar{f}_i \cap \bar{f}_0 = \phi)$ where *1≤i≤n*. There is at least one frame that does not overlap with the first frame.

Constraint 2. $\forall i($ ¤*(δi)* Λ ¤*(*$\bar{\delta}_i$*))* where *1≤i≤n*. The new regions in new frames do not have moving objects.

Lemma 1. For two regions R_i and R_j, if ¤*(Ri)* Λ ¤*(Rj)* → ¤*(Ri U Rj)* for the union of R_i and R_j.

Lemma 2. Using Lemma 1, for a region δ_i, ¤*(δi)* Λ ¤*(Si-1)* → ¤*(Si)*. In other words, if the sprite S_{i-1} before blending f_i does not have any moving objects, the sprite S_i after blending f_i will not have any moving objects. Also note that $\delta_i \cap S_{i-1} = \emptyset$.

Proof by induction for region A. Let A_k represent the area A in Fig. 10 by generating sprite using frames f_1 through f_k. B is initially equivalent to f_0. According to conservative sprite a frame may contribute at most 4 regions: R_1, R_2, R_3, and R_4. Let $R = (((R_1 \cup R_2) \cup R_3) \cup R_4)$. Our claim is that if R_1, R_2, R_3, and R_4 are free of objects for each frame, then the region A in Fig. 10 will not contain objects, since the sprite is expanded by using these 4 regions only.

Basis: The region B is equivalent to frame f_0. The next frame, f_1, will contribute at most 4 regions: $R = R_1 \cup R_2 \cup R_3 \cup R_4$ is free of objects. Therefore, A_1 cannot have any objects.

Assume that A_k does not contain any object ($1 \le k \le m$). We have a new frame f_{k+1} to be warped and blended onto A_k. We would like to induce that A_{k+1} does not contain any objects (i.e., $\neg(A_{k+1})$ given $\neg(A_k)$). We know that $\delta_{k+1} = \hat{f}_{k+1} - A_k$; thus we can infer $\neg(\delta_{k+1})$ using *constraint 2*. Hence, we have i) $\neg(\delta_{k+1})$ and ii) $A_{k+1} = A_k \cup \delta_{k+1}$. Using *Lemma 1*, $\neg(A_k) \wedge \neg(\delta_{k+1}) \rightarrow \neg(A_k \cup \delta_{k+1})$. Therefore, $\neg(A_k) \wedge \neg(\delta_{k+1}) \rightarrow \neg(A_{k+1})$.

Proof for equivalence of assertive and conservative sprites.

Lemma 3. The sprite that is generated using two sequential frames f_i and f_{i+1} by utilizing conservative sprite is equivalent to the sprite that is generated by the same sequential frames in the reverse order by utilizing assertive sprite (Fig. 11).

Lemma 4. The conservative sprite that is generated for a sequence from f_n to f_0 is equivalent to assertive sprite that is generated in the reverse order of frames.

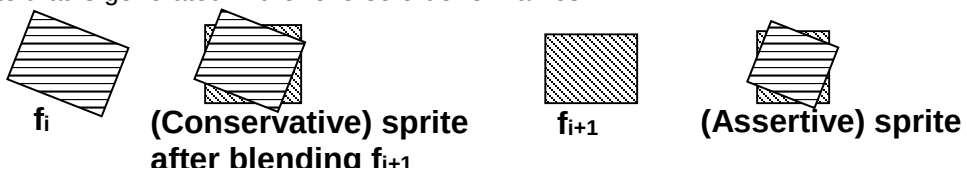


Fig. 11. Equivalence of conservative sprite and assertive sprite.

Basis is provided by *Lemma 3*. Assume that the conservative sprite for f_1 to f_k is equivalent to assertive sprite for f_k to f_1.We need to show that if a new frame f_{k+1} needs to be blended, the conservative sprite and assertive sprite should be equivalent. Note that f_{k+1} is the first frame for assertive sprite. In conservative sprite, f_{k+1} will not affect the previously generated sprite but new unvisited regions of f_{k+1} will be added to the sprite (based on definition of conservative sprite). New unvisited regions will also have a similar case in assertive sprite. The overlapping area would be overwritten by the other frames since f_{k+1} is the first frame for the assertive sprite. The only difference is the new regions which are not covered in any of the other frames. We should note that each frame corresponds to same area on each sprite since the global motion estimation is the same for each sprite.

Proof for region B. We are going to prove that region B can be generated without objects using the previous proof for region A and Lemma 4.

Constraint 1 states that there is at least one frame (e.g., f_m) that does not intersect with frame f_0. If a conservative sprite is generated starting from f_m to f_0, region B will not have any moving objects based on the previous proof for region A. This sprite is equivalent to assertive sprite for frames starting f_0 to f_m using *Lemma 4*.

Since both regions A and B can be generated without objects, the final sprite is free of moving objects. The constraints can be relaxed further but this complicates the proof. Actually, absence of moving objects at borders is not the absolute requirement. Moving objects at the borders are allowed as long as they do not appear in the new visible areas (regions) when compared to previous frames.

3.4 Discussion and Optimization

At first sight, it might be thought that generating two sprites at a time would be costlier than traditional sprite generation based on object segmentation. As mentioned at the beginning of this section, the three steps of sprite generation are GME, warping, and blending. Our algorithm does not add any significant complexity due to:

- GME is the most expensive component of sprite generation [MPEG4 Software; Nagaraj 2001]. In our method, GME is only performed once using conservative sprite. The obtained parameters are used for assertive sprite. In terms of GME complexity, there is no additional cost in our algorithm.

- Warping and blending need to be applied twice since there are two sprites in our case. Note that assertive sprite only adds new pixels to the sprite. Hence, bilinear interpolation only needs to be applied for regions corresponding to new areas in this sprite. In [Nagaraj 2001], it is assumed that only 10% of pixels correspond to new areas for a frame. Our experience is compliant with this assumption.
- For fusion on conservative sprite, the pixel values of the assertive sprite only update the region for the first frame in the sequence. Instead of maintaining two sprites, the warping and blending for assertive sprite just needs to be performed if a frame is overlapping with the first frame in the sequence. Once pixel values are obtained for the region of first frame on conservative sprite, the warping and blending for assertive sprite generation can be ignored.
- Fusion can be performed whenever a frame does not intersect the first frame. After the fusion, the sprite evolves like a conservative sprite.

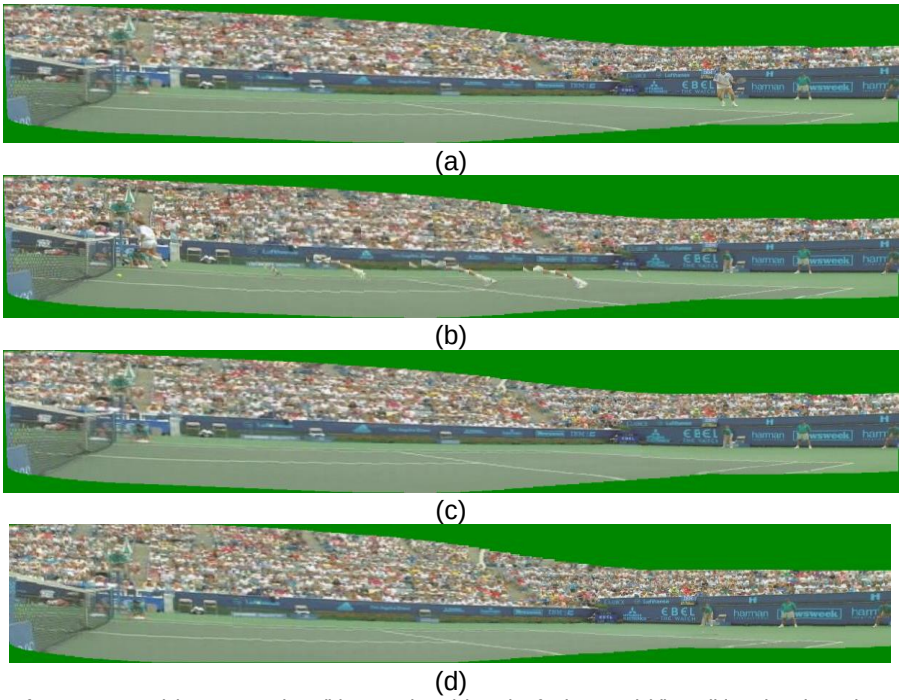
4. EXPERIMENTS

We developed a database of a variety of videos. These videos are accessible at http://sprite.cs.uah.edu/. We have implemented our sprite generation system using Microsoft Visual C# under Windows. We have experimented our technique on a variety of sequences including stefan and coastguard. If the test sequences have noise at the frame borders (i.e., black border on the sides of a frame), we try to eliminate them during sprite generation. We discuss our results on these videos.

In our experiments mentioned in this paper, we used 300 frames for Stefan, 300 frames for Exploring Turkey videos, 181 frames for coastguard video, 214 frames for frame for tracking boat video, and 117 frames for tracking field video. Typically, the number of frames is between 100 and 500.

4.1 EXPERIMENTS ON TRADITIONAL TEST SEQUENCES

4.1.1 *Stefan Video.* Without careful analysis of the frames of stefan sequence, the stefan video would be considered as a tracking video. In other words, the cameraperson tries to locate the object



(a) (b) (c) (d)

Fig. 12. Sprites for stefan sequence (a) conservative, (b) assertive, (c) sprite fusion, and (d) traditional sprite using segmentation masks

(tennis player) in the middle of a frame. Actually, the cameraman was not successful in achieving this goal. This becomes obvious in the assertive sprite where the sprite has parts of the object when the cameraman was not successful (Figure 12 (b)). Hence, the *stefan* sequence is partially-tracking video rather than a

tracking video. It has also been noticed that the object in the initial frame does not appear in the assertive sprite.

The conservative sprite of stefan sequence is generated as expected (Figure 12 (a)). The conservative sprite as an artifact only has the initial frame in the sequence. This also means that no object is visible in new areas (after camera motion) in the scene. The final sprite is generated by fusing assertive sprite on conservative sprite (Figure 12 (c)).

To compare our result with traditional sprite by using traditional sprite generation, we also provide this sprite in Figure 12 (d). Visually, our fused sprite is almost the same as the regular sprite generated by using segmentation masks. In the fused sprite, it is possible to see discontinuities in the lines. On the other hand, the lines of the field sometimes are blurred in the traditional sprite.

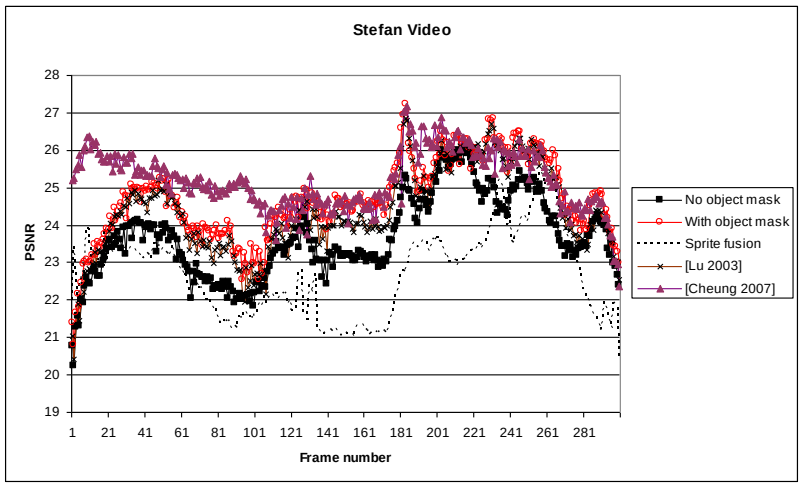


Fig. 13. Our PSNR results for Stefan video.

Although PSNR is not always a good indication of human visual judgment, we provide PSNR results for Stefan. Stefan is one of the most common examples for sprite generation. We provide the PSNR values for the sprite fusion, the traditional sprite without using segmentation masks, and the traditional sprite using available segmentation masks for the stefan sequence. The average PSNR values for sprite fusion, traditional sprite w/o segmentation masks, and traditional sprite with segmentation masks are 22.69, 23.67, and 24.68, respectively (Figure 13). We implemented the algorithms suggested by [Lu 2003] and [Cheung 2007] based on our sprite generation. We get average PSNR values 24.37 (for [Lu 2003]) and 25.22 (for [Cheung 2007]). The first one applies rough object segmentation whereas the second one requires object masks. The average PSNR for stefan is mentioned as 22.315 in [Cheung 2007]. This shows that just using the final quality score is not enough that a method is better than the other. This indicates that our basic sprite generation algorithm performs better than the other previously proposed techniques.

4.1.2 Coastguard Video. Although there are many experiments performed on coastguard sequence, this sequence has the least satisfactory results. The problems in the coastguard sprite can be listed as follows: a) the sprite for the coast shore generated only, b) the shadows of the boat cannot be removed from the water, and c) the texture of the water is removed during averaging. To the best of our knowledge, the only research that was able to generate the coastguard sequence properly was presented in [Hsu 2004]. However, their method relies on object segmentation.

We are going to provide the results for coastguard video in two parts. We firstly, investigate the sequence from frame 120 through frame 300. In this part, the coastguard boat is the only moving object in the scene. This section of the video can be considered as a tracking video. Figure 14 displays the conservative, assertive, and sprite fusion. In this case, the sprite is generated as in Figure 14 (c). However, this introduced an artifact. This is due to the changing illumination towards the end of the sequence. However, this problem can be resolved with image enhancement techniques like gamma correction (Figure 14 (d)). Figure 14 (e) shows the results of traditional sprite generation method with artifacts.

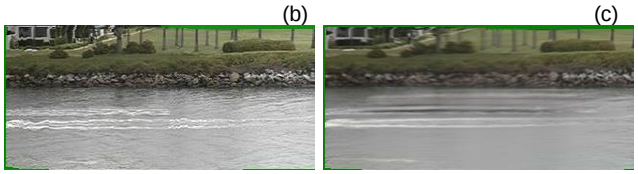


Fig. 14. Sprites for coastguard sequence (from frame 120 through frame 300) (a) conservative, (b) assertive, (c) sprite fusion, (d) corrected sprite, and (e) traditional sprite.

In the complete video of the coastguard sequence, the camera (pans-left) tracks a black boat as another white boat enters the scene. As the white boat becomes visible, the camera (pans-right) starts tracking the white boat. The coastguard sequence is a type of complex video. The major problem with sprite generation for coastguard sequence is the black pattern on the bottom of the white boat. Although the boat moves, the pattern stays stable because of the continuity of the pattern. To the best of our knowledge, the traditional sprite generation methods that rely on temporal integration cannot cope with this since the pattern is the dominant area for that region (in other words, the pattern of the boat is observed more than the background water for some regions in the scene). In the literature, sprites for coastguard usually cover the land and not the water due to this.

It is possible to overcome this problem by enhancement on sprite fusion. The conservative and assertive sprites for coastguard sequence are shown in Figures 15 (a) and (b), respectively. Since this is a complex video, applying the sprite fusion on assertive or conservative sprite would not yield the correct sprite. In this case, we match the pixels of both sprites and choose the one that is closer to the surrounding pattern. We then get the sprite in Figure 15 (c). This method covers some of the complex videos. As future work, this pattern comparison method needs to be validated against different types of sequences. Here, we just show that sprite fusion is still possible with some enhancement.

In the literature, it is mentioned global motion estimation is the most time-consuming process [Nagaraj 2001]. However, optimized algorithms are developed for global motion estimation. For example, average time to compute global motion estimation of a frame of a tennis video in CIF format is 9ms using pixel subsampling method [Alzoubi 2008]. The average time to process a frame in CIF format is 12-15ms [Richter 2001]. We compute the time taken for each component of our sprite generation algorithm. The major difference in our algorithm is the blending part of the sprite generation algorithm. In our algorithm, we used a compact sprite. Whenever camera pans, all pixels on the sprite should be shifted to right. Alternate way is to use a large sprite and this shifting can be avoided. Blending part is the time to blend pixels on these sprites. Average time taken for blending for Stefan video is 21.39ms with traditional blending and 24.19ms with sprite fusion. On the other hand, average time taken for blending for coastguard video is 19.87ms with traditional blending and 18.39 with sprite fusion. Since coastguard video can be represented with translational motion, its blending is faster than the blending of Stefan video. For Stefan video, sprite fusion added 2.8ms on the average. On the other hand, for coastguard video, sprite fusion finished 1.5ms earlier than the traditional blending. Our fusion method does not any major burden on the system. Our algorithms are used for debugging purposes now. Our code can further be optimized for performance.

We maintain 3 sprites in the memory: one for assertive sprite, one for conservative sprite, and one sprite for global motion estimation. For fusion, conservative and assertive sprites are fused onto a sprite. After fusion, conservative and assertive sprites are not needed. The memory is allocated dynamically and initially their sizes are equivalent to the size of a frame. In addition, we maintain the previous frame and current frame in the memory.

Fig. 15. Sprites for coastguard sequence (a) conservative, (b) assertive, and (c) sprite fusion based on pattern comparison.

4.2 CHALLENGING VIDEOS and FUTURE WORK

In this section, we introduce 3 types of videos where sprite generation is difficult but still can be achieved at a level. We introduce the problems and show sprites that could be generated.

4.2.1 *Tracking Boat Video.* This sequence is stored as tracking boat in our data set. Sample frames of this sequence are shown in Figure 16. The earlier frames appear on the right-hand side of the figure. In this sequence, there is a large boat that moves to the left. The camera tracks this boat.

Fig. 16. The tracking boat sequence.

Firstly, this is not an easy sequence to generate a sprite since there is a large object in the scene. The background is only visible on the left-hand side of the frames. It is also difficult to perform GME due to static patterns in the scene and large moving object with the camera (Figure 17).

There are two separate motions in this sequence: the static motion of the boat with respect to camera and the camera motion that can be determined from the shore. After enhancing our sprite generation method, we get the sprite in Figure 18. We deliberately maintained the first frame to see the complete picture. We still work on this type of videos.

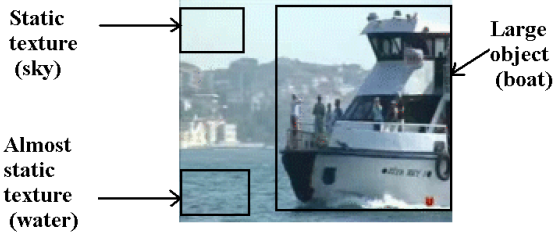


Fig. 17. Difficulties for sprite generation

Fig. 18. The sprite for tracking boat.

4.2.2 *Tracking Video -2*. Tracking video-2 is a sequence for tracking athletes in a competition. Sample frames from the sequence are shown in Figure 19. It is also difficult to generate sprite for such a video. The problems for this sequence are: a) static logos, b) constant occlusion of the middle of the scene by athletes, and c) almost static appearance of the track. Figure 20 shows these problems. After enhancing our sprite generation method, we get the sprite in Figure 21. Although the accuracy of the sprite is low, we were able to generate a sprite for the sequence. We still work on this type of videos.

Fig. 19. The tracking athletes sequence.

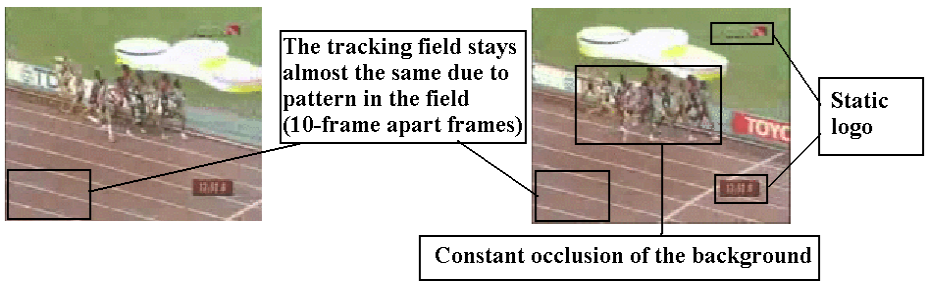


Fig. 20. The problems with tracking athletes sequence.

Fig. 21. The sprite for tracking athletes.

4.2.3 *Exploring Turkiye-3 Video*. Exploring Turkiye-3 video is a sequence with many objects in the scene. The difficulty for this sequence is the quantity of moving objects. Sample frames from the sequence are shown in Figure 22. The sprite obtained by sprite fusion is shown in Figure 23. The artifacts at the bottom of the sprite are due to noise at frame borders.

Fig. 22. The sample frames for exploring Turkey.

Fig. 23. The sprite for exploring Turkey.

4.3 SUMMARY

Firstly, we showed that sprite fusion produces acceptable results for traditional test sequences such as *stefan* and *coastguard*. In addition, sprite fusion can still produce good results even in the presence of many objects. We show that sprite fusion can generate acceptable results for challenging videos such as Exploring Turkey – 3 video. We introduced two examples of videos (tracking boat and tracking video-2) where sprite generation is very difficult. We show that the sprite can be generated by enhancing our method. We still work on enhancement so that our improvements are applicable to any video in this category. We tried to maintain our methods not computationally intensive. Our results indicate good and promising results for future sprite generation methods.

5. CONCLUSION AND FUTURE WORK

In this paper, we have provided some of the issues for sprite generation. Then, we focused on the domain of videos. We have classified the types of videos and then mentioned which classes of video would benefit from our approach. We introduced sprite fusion method for sprite generation. Sprite fusion can be used in tracking and partially-tracking videos. The advantage of sprite fusion is that it does not rely on object segmentation or determination of object pixels. Moreover, it has returned comparatively good results with respect to sprite generation methods that rely on object segmentation masks. We also showed how it is useful for generating the sprite on videos that are difficult for sprite generation. It should also be noted that our algorithms are not computationally complex since they do not require object segmentation. Our target in this research is to extend the domain of videos where the sprites can be generated. As the domain of these videos increases, the accuracy of the sprites can also be improved. After satisfactory results are obtained in uncompressed domain, sprite generation should be performed in compressed domain by benefiting from Discrete Cosine and Discrete Wavelet Transforms.

REFERENCES

Alzoubi, H and Pan, W. D., 2008. Fast and accurate global motion estimation algorithm using pixel subsampling. *Inf. Sci.* 178, 17 (September 2008), 3415-3425.

ASIF, M. AND SORAGHAN, J. J. 2008. MPEG-7 Motion Descriptor Extraction for Panning Camera Using Sprite Generated. In *Proceedings of the 2008 IEEE Fifth international Conference on Advanced Video and Signal Based Surveillance* (September 01 - 03, 2008). AVSS. IEEE Computer Society, Washington, DC, 60-66

ATREY, P.K. HOSSAIN, M.A. SADDIK A.E. AND KANKAHALLI, M.S. 2010. MULTIMODAL FUSION FOR MULTIMEDIA ANALYSIS: SURVEY. IN MULTIMEDIA SYSTEMS, VOLUME 16, ISSUE 6 (2010), PAGE 345-379.

AZZARI, P. DI STEFANO, L. AND BEVILACQUA, A. 2005. An effective real-time mosaicing algorithm apt to detect motion through background subtraction using a PTZ camera. In *Advanced Video and Signal Based Surveillance, (September 2005). AVSS 2005. IEEE Conference on* , vol., no., pp. 511-516, 15-16.

AYGÜN, R. S. AND ZHANG, A. 2002. Reducing blurring-effect in high resolution mosaic generation. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, vol.2, IEEE Computer Society, Washington, DC, 537-540.

AYGÜN, R. S. AND ZHANG, A. 2004, Integrating virtual camera controls into digital video. In *Multimedia and Expo, 2004. ICME '04.(June 2004)IEEE Int. Conference on* , vol.3, no., pp.1503-1506 Vol.3, 30-30.

AYGUN, R.S. AND ZHANG, A. 2004. Sprite pyramid for videos and images having finite-depth scenes. In *Multimedia and Expo, 2004. ICME '04.(June 2004) IEEE International Conference on* , vol.2, no., pp.795-798 Vol.2, 30-30.

CHEN, Y. AND AYGUN. R.S. 2010. Synthetic Video Generation for Evaluation of the sprite generation. In *International Journal of Multimedia Data Engineering & Management.vil. , no. 2, pp. 34-61, April-June 2010*

CHEN, SY. CHEN, CY. HUANG, YW. AND CHEN, LG. 2002. Multiple sprites and frame skipping techniques for sprite generation with high subjective quality and fast speed. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE Int. Conference on* , vol.1, IEEE Computer Society, Washington, DC, 785-788.

CHEN, L., LAI, Y., AND LIAO, H. 2006. Video Scene Extraction Using Mosaic Technique. In *Proceedings of the 18th international Conference on Pattern Recognition - Volume 04* (August 20 - 24, 2006). ICPR. IEEE Computer Society, Washington, DC, 723-726.

CHEUNG, H.-K. AND SIU W.-C. 2007. Robust global motion estimation and novel updating strategy for sprite generation. In *Image Processing, IET* (March 2007), vol.1, no.1, pp.13-20.

CHERNG, DC. AND CHIEN SY. 2007. Video Segmentation with Model-Based Sprite Generation for Panning Surveillance Cameras. In *Circuits and Systems,(* May *2007). ISCAS 2007. IEEE International Symposium on* , IEEE Computer Society, Washington, DC, 27-30.

CHEUNG, H.K. AND SIU, W.C. 2002. Fast global motion estimation for sprite generation. In *Circuits and Systems, 2002. ISCAS 2002. IEEE Int. Symposium on*, vol.3, IEEE Computer Society, Washington, DC.

CHEUNG, H.K. SIU, W.C. AND FENG D. 2008. New Block-Based Motion Estimation for Sequences with Brightness Variation and Its Application to Static Sprite Generation for Video Compression. In *Circuits and Systems for Video Technology (April 2008),* vol.18, IEEE Computer Society, Washington, DC, 522-527

COORG, S. AND TELLER, S. 2000. Spherical mosaics with quaternions and dense correlation, International Journal of Computer Vision 37 (3) (2000) 259–273.

DASU, A.R. AND PANCHANATHAN, S. 2004. A wavelet-based sprite codec. *Circuits and Systems for Video Technology (*February 2004*), IEEE Transactions on* , vol.14, no.2, pp. 244-255.

DESHPANDE, A. AND AYGUN, R.S. 2009. Motion-based video classification for sprite generation. Database and Expert Systems Applications, International Workshop on, pp. 231-235, 2009 20th International Workshop on Database and Expert Systems Application, 2009

DUFAUX, F. AND KONRAD, J. 2000. Efficient, robust, and fast global motion estimation for video coding, IEEE Transactions on Image Processing 9 (3) (2000) 497–501.

FARIN, D. AND DE WITH, P. H. N. 2006. Enabling arbitrary rotational camera motion using multisprites with minimum coding cost., IEEE Trans. Circuits Syst. Video Techn. 16 (4) (2006) 492–506.

FRAUNHAUFER, http://www.iis.fraunhofer.de/amm/download/mpeg4 (June 2009).

GRAMMALIDIS, N. BELETSIOTIS, D. AND STRINTZIS, M.G. 1999. Multi View sprite generation and coding. In *mage Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on* , vol.2, IEEE Computer Society, Washington, DC,477-481.

GEYS, H. AND GOOL, LUC VAN, 2006. On-line, interactive view synthesis and augmentation, Signal Processing: Image Communication 21 (9) (2006) 709–723.

H264. Iso/iec 14496-10:2003, information technology: Coding of audio-visual objects - part 2, also itu-t recommendation h.264 advanced video coding for generic audiovisual services.

H. 265. http://www.h265.net (2009)

HSU, C.-T. TSAN Y.-C. TSAN (2004) Mosaics of video sequences with moving objects, Signal Processing: Image Communication, Volume 19, Issue 1, January 2004, Pages 81-98, ISSN 0923-5965, DOI: 10.1016/j.image.2003.10.001.

IRANI, M. AND ANANDAN, P. 1998. Video indexing based on mosaic representations, in: Proceedings of IEEE, 1998, pp. 905–921.

KRUTZ, A. GLANTZ, A. HALLER, M. DROESE, M. AND SIKORA, T. 2008. Multiple Background Sprite Generation Using Camera Motion Characterization for Object-Based Video Coding. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video* (May *2008)*, vol., no., pp.313-316, 28-30.

KRUTZ, A. GLANTZ, A. SIKORA, T. NUNES, P. AND PEREIRA, F. 2008. Automatic object segmentation algorithms for sprite coding using MPEG-4. In *ELMAR, (September* 10 – September 12, *2008. 50th International Symposium* , vol.2, no., pp.459-462.

KRUTZ, A. FRATER, M. KUNTER, M. AND SIKORA, T. 2006. Windowed Image Registration for Robust Mosaicing of Scenes with Large Background Occlusions. In *Image Processing, (October 2006) IEEE International Conference on*, vol., no., pp.353-356, 8-11.

KUNTER, M. KREY, P. KRUTZ, A. AND SIKORA, T. 2008. Extending H.264/AVC with a background sprite prediction mode. In *Image Processing, (October2008). ICIP 2008. 15th IEEE International Conference on*, vol., no., pp.2128-2131, 12-15.

LAI, J., KAO, C., AND CHIEN, S. 2009. Super-resolution sprite with foreground removal. In *Proceedings of the 2009 IEEE international Conference on Multimedia and Expo* (New York, NY, USA, June 28 - July 03, 2009).

LEE M-C; CHEN W-G; LIN, C.B.; CHUANG GU; MARKOC, T.; ZABINSKY, S.I.; SZELISKI, R., "A layered video object coding system using sprite and affine motion model," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.7, no.1, pp.130-145, Feb 1997
LU, Y., GAO, W., AND WU, F. 2001. Fast and Robust Sprite Generation for MPEG-4 Video Coding. In *Proceedings of the Second IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia information Processing* (October 24 - 26, 2001). H. Shum, M. Liao, and S. Chang, Eds. Lecture Notes In Computer Science, vol. 2195. Springer-Verlag, London, 118-125.
LU, Y. GAO, W AND WU, F. 2001. Sprite generation for frame-based video coding. In *Image Processing, 2001. Proceedings (2001). 2001 International Conference on,* vol.1, no., pp.473-476 vol.1, 2001.
LU, Y., GAO, W., AND WU, F. 2003. Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques, IEEE Trans. Circuits and Systems for Video Technology 13 (5) (2003) 394–405.
MARZOTTO, R. FUSIELLO, A. AND MURINO, V. 2004. High resolution video mosaicing with global alignment. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the (27 June-2 July 2004) IEEE Computer Society Conference on* , vol.1, no., pp. I-692-I-698 Vol.1.
MPEG4-2, Iso/iec 14496-2:2004, information technology: Coding of audio-visual objects - part 2.
MPEG4 SOFTWARE, Iso/iec 14496-7:2001, information technology: Coding of audio-visual objects - part 7: Optimized software for mpeg-4 visual tools.
NAGARAJ, R. C., DASU, A. R., AND PANCHANATHAN, S. 2001. Complexity analysis of sprites in mpeg-4, in: Proc. SPIE Vol. 4313, 2001, pp. 69–73.
OSTERMANN, J., BORMANS, J., LIST, P., MARPE, D., NARROSCHKE, M., PERREIRA, F., STOCKHAMMER, T. AND WEDI, T. 2004. Video coding with h.264/avc: tools, performance, and complexity, IEEE Circuits and Systems Magazine 4 (1) (2004) 7–28.
PARIKH, P. AND JAWAHAR, C. V. 2007. Enhanced Video Mosaicing using Camera Motion Properties. In *Proceedings of the IEEE Workshop on Motion and Video Computing* (February 23 - 24, 2007). WMVC. IEEE Computer Society, Washington, DC, 26
PELEG, S., ROUSSO, B., RAV-ACHA, A., AND ZOMET, A. 2000. Mosaicing on adaptive methods, IEEE Trans. on Pattern Analysis and Machine Intelligence 22 (10) (2000) 1144–1154.
PRODYS, http://www.prodys.com/ (June 2009).
Richter, H., Smolic, A., Stabernack, B. and Müller, E., Real Time Global Motion Estimation for an MPEG-4 Video Encoder, Proc. PCS'2001, Picture Coding Symposium, 25.-27. April 2001, Seoul, Korea.
SALEMBIER, P., PUJOL, O., AND GARRIDO, L. 1998. Connected operators for sprite creation and layered representation of image sequences, in: IV European Signal Processing Conference, 1998, pp. 2105–2108.
SHEN, Y. AND ZHANG, L. 2004. A Novel Method of Sprite Generation Based on Pixel Similarity. In *Proceedings of the Third international Conference on Image and Graphics* (December 18 - 20, 2004). ICIG. IEEE Computer Society, Washington, DC, 560-563.
SIKORA, T. 1997. The mpeg-4 video standard verification model, IEEE Trans. Circuits Syst. Video Technology 7 (1997) 19–31.
SMOLIC, A., SIKORA, T., AND OHM, J.-R. 1999. Long-term global motion estimation and its application for sprite coding, content description and segmentation, IEEE Transactions on Circuits and Systems for Video Technology 9 (8) (1999) 1227–1242.
SMOLIC, A. AND OHM, J.-R. 2000. Robust global motion estimation using a simplified m-estimator approach, in: Proc. ICIP2000, IEEE International Conference on Image Processing, 2000.
SNOEK, C. G., WORRING, M., AND SMEULDERS, A. W. 2005. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th Annual ACM international Conference on Multimedia* (Hilton, Singapore, November 06 - 11, 2005). MULTIMEDIA '05. ACM, New York, NY, 399-402.
STEEDLY, D. PAL, C. AND SZELISKI, R. 2005. Efficiently registering video into panoramic mosaics. *Computer Vision, 2005. ICCV(October2005).* In *Tenth IEEE Int. Conf. on* , vol.2, no., pp.1300-1307 Vol. 2, 17-21
SZELISKI, R., AND SHUM, H.-Y. 1997. Creating full view panoramic image mosaics and environment maps, in: Computer Graphics Proceedings, Annual Conference Series, 1997, pp. 251–258.
SZELISKI, R. 2006. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.* 2, 1 (Jan. 2006), 1-104.
TAUBMAN, D. AND MARCELLIN, M. 2002. JPEG2000 - Image Compression Fundamentals, Standards and Practice, chapter 10, Kluwer Academic Publishers, 2002.
TEODOSIO, L. AND BENDER, W. 1993. Salient video stills: content and context preserved. In *Proceedings of the First ACM international Conference on Multimedia* (Anaheim, California, United States, August 02 - 06, 1993). MULTIMEDIA '93. ACM, New York, NY, 39-46.
TEODOSIO, L. AND BENDER, W. 2005. Salient stills. *ACM Trans. Multimedia Comput. Commun. Appl.* 1, 1 (Feb. 2005), 16-36.
TO, L. T. 2005. Video object segmentation using phase-based detection of moving object buondaries, Ph.D. thesis, University of New South Wales (2005).
YE, G. PICKERING, M. FRATER, M. AND ARNOLD, J. 2005. A robust approach to super-resolution sprite generation. *Image Processing, (September 2005). ICIP 2005. IEEE International Conference on* , vol.1, IEEE Computer Society, Washington, DC, 11-14.
YE, G. WANG Y. XU, J. HERMAN, G. AND ZHANG, B. 2008. A practical approach to multiple super-resolution sprite generation. In *Multimedia Signal Processing,(October 2008) IEEE 10th Workshop on* , IEEE Computer Society, Washington, DC, 70-75, 8-10.
ZHU, Z., XU, G., RISEMAN, E. M., AND HANSON, A. R. 1999. Fast Generation of Dynamic and Multi-Resolution 360-Degree Panorama from Video Sequences. In *Proc. of the IEEE int. Conf. on Multimedia Computing and Systems - Volume 2* (June 07 - 11, 1999). ICMCS. IEEE Computer Society, Washington, DC, 9400.
ZOGHLAMI, I., FAUGERAS, O., AND DERICHE, R. 1997. Using geometric corners to build a 2d mosaic from a set of images, in: IEEE Int. Conference on Computer Vision and Pattern Recognition, 1997, pp. 420–425.

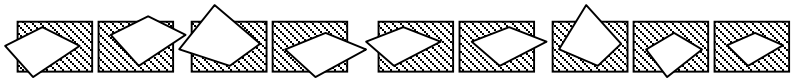
APPENDIX

A.1 POSSIBLE OVERLAPPINGS FOR TWO FRAMES

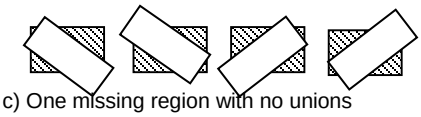
To generate all possible cases, there are three operations: *union, presence,* and *absence*. The regions may unite with neighboring regions or that may not exist at all. We are going to represent union with U,

absence with -, and presence with +. If a region appears in a union, it is automatically present. Below, R_1, R_2, R_3, and R_4 represent 4 corner regions shown in Fig. 9. For Fig. A.1, we are going to provide the representation for the left-most (lm) and the right-most (rm) for each sub-figure: **a)** (lm) $(R_1UR_4)+R_2+R_3$, (rm) $(R_1UR_2)+(R_3UR_4)$ **b)** (lm) $(R_1UR_3UR_4)+R_2$, (rm) $(R_1UR_2UR_3UR_4)$ **c)** (lm) $R_1+R_2+R_4-R_3$, (rm) $(R_1UR_2UR_3)-R_4$ **d)** (lm) $(R_1UR_4)+R_2-R_3$, (rm) $(R_1UR_2)+R_3-R_4$ **e)** (lm) $(R_1UR_2UR_4)-R_3$ (rm) $(R_1UR_2UR_3)-R_4$ **f)** (lm) $R_2+R_4-R_1-R_3$ (rm) $R_1+R_4-R_2-R_3$ **g)** (lm) $(R_1UR_2)-R_3-R_4$, (rm) $(R_1UR_4)-R_2-R_3$ **h)** (lm) $R_1-R_2-R_3-R_4$, (rm) $-R_1-R_2-R_3-R_4$.

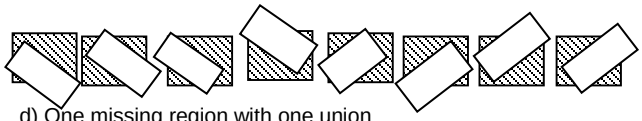
a) Left side: only one union with the neighboring region; Right side: two unions



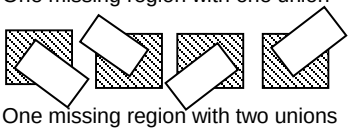
b) 3 region unions (first 4); all regions union (the last 5)



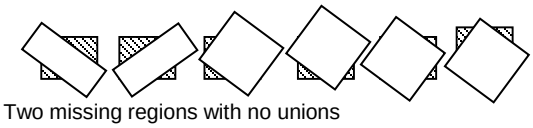
c) One missing region with no unions



d) One missing region with one union

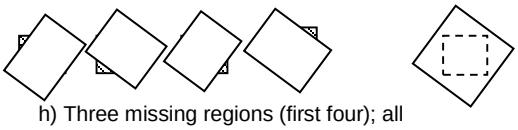


e) One missing region with two unions



f) Two missing regions with no unions

g) Two missing regions with unions



h) Three missing regions (first four); all regions are missing (the last one)

Fig. A.1. Overlappings between two frames.

A.2 Fusing Assertive Sprite onto conservative Sprite

We provide the algorithm for fusing assertive sprite onto conservative sprite.

Algorithm 2. Algorithm for fusing assertive sprite onto conservative sprite
// IN: Assertive sprite AS_{n-1}
// IN: Conservative sprite CS_{n-1}
// IN: frames f_n and f_{n-1} of the video
// OUT: Fused sprite FSA on conservative sprite
apply the necessary shift operations on AS_{n-1} and CS_{n-1}
copy CS_{n-1} to FSA

```
(minx,miny) ← (shiftx, shifty) // the top-left coordinate of the 1st frame on CS
for i = miny to (miny+height) do
  for j = minx to (minx+width) do
    FSA(i, j) = AS(i, j)
  end for
end for
```